Научно-исследовательский практикум

РАЗНОЕ

Операторы

УНАРНЫЕ

i++

--j

++k

!a - отрицание

…

БИНАРНЫЕ

a + b

c / d

f % h

a = b

a > b

c == d

…

Тренарный оператор

Единственный оператор с 3 операндами в языке

(выражение) ? *(если истинно)* : *(если ложно)*

Например

```
int a = flag? 1 : 0;
```

- Оператор хорошо оптимизирован
- Но снижает читаемость кода

Пример использования

```
int main(int argc, char *argv[])
{
bool flag = true;
if (!flag)
cout << "true" << endl;
else
cout << "false" << endl;

int a = flag? 1 : 0;
cout << flag << " " << a << endl;

for (int i=0; i<10; i++)
cout << i << ((i%2) ? " odd" : " even") << endl;

return 0;
}
```

Что получилось

```
Starting /Users/amakashov/projects/build-stuff
false
1  1
0 even
1 odd
2 even
3 odd
4 even
5 odd
6 even
7 odd
8 even
9 odd
/Users/amakashov/projects/build-stuff-Desktop-
```

Приоритет операторов

Приоритет	Операция	Описание	Ассоциативность
1	::	Область видимости	Слева направо
2	a++ a--	Постинкремент и постдекремент	
	()	Вызов функции	
	[]	Обращение к массиву по индексу	
	.	Выбор элемента по ссылке	
	–>	Выбор элемента по указателю	

Приоритет операторов

3	++a --a	Прединкремент и преддекремент	Справа налево
	+ −	Унарный плюс и минус	
	! ~	Логическое НЕ и побитовое НЕ	
	(*type*)	Приведение к типу type	
	*	Indirection (разыменование)	
	&	Адрес	
	sizeof	Размер	
	new, new[]	Динамическое выделение памяти	
	delete, delete[]	Динамическое освобождение памяти	

Приоритет операторов

4	.* ->*	Указатель на член	Слева направо
5	* / %	Умножение, деление и остаток	
6	+ −	Сложение и вычитание	
7	<< >>	Побитовый сдвиг влево и вправо	
8	< <=	Операции сравнения < и ≤	
	> >=	Операции сравнения > и ≥	
9	== !=	Операции сравнения = и ≠	
10	&	Побитовое И	
11	^	Побитовый XOR (исключающее ИЛИ)	

Приоритет операторов

12	|	Побитовое ИЛИ (inclusive or)	Слева направо
13	&&	Логическое И	
14	||	Логическое ИЛИ	
15	?:	Тернарное условие	Справа налево
	=	Прямое присваивание (предоставляемое по умолчанию для C++ классов)	
	+= −=	Присвоение с суммированием и разностью	
	*= /= %=	Присвоение с умножением, делением и остатком от деления	
	<<= >>=	Присвоение с побитовым сдвигом слево и вправо	
	&= ^= |=	Присвоение с побитовыми логическими операциями (И, XOR, ИЛИ)	

static_cast и dynamic_cast

```
class Base
{
public:
virtual void Function() {cout << "Base function" << endl;}
};

class Derived : public Base
{
public:
virtual void Function() {cout << "Derived function" << endl;}
void NonVirtualFunction() {cout << "Derived non-virtualfunction" << endl;}
};
```

main

```
int main(int argc, char *argv[])
{
Base* ptr = new Derived;
ptr->Function();
Derived* derPtr = static_cast<Derived*>(ptr);
derPtr->NonVirtualFunction();
return 0;
}
```

```
Starting /Users/amakashov/projects/buil
Derived function
Derived non-virtual function
/Users/amakashov/projects/build-stuff-[
```

Чуть изменим классы...

```
class Base
{
public:
virtual void Function() {cout << "Base function" << endl;}
};

class Derived : public Base
{
public:
   virtual void Function() {cout << "Derived function" << endl;}
   void NonVirtualFunction() {cout << "Value " << value << endl;}
protected:
   int value = 100;
};
```

…и mian()

```
int main(int argc, char *argv[])
{
    Base* ptr = new Base;
    ptr->Function();
    Derived* derPtr = static_cast<Derived*>(ptr);
    derPtr->NonVirtualFunction();
    return 0;
}
```

```
Starting /Users/amakashov/projects/bui
Base function
Value -1373372406
/Users/amakashov/projects/build-stuff-
```

dynamic_cast

```
int main(int argc, char *argv[])
{
Base* ptr = new Base;
ptr->Function();
Derived* derPtr = dynamic_cast<Derived*>(ptr);
if (derPtr)
    derPtr->NonVirtualFunction();
else
    cout << "Unable to cast pointer" << endl;
return 0;
}
```

```
Starting /Users/amakashov/projects/bui
Base function
Unable to cast pointer
/Users/amakashov/projects/build-stuff-
```